a data object stored within the directory file and between the start address and the end address;

pointer data associated with the data object stored within the directory file and between the start address and the end address; and,

a continuous block of available memory between the last stored pointer data and the data object location indicated by the last stored pointer data, the continuous block of available memory for storing therein of pointer data and data objects.

Brief Description of the Drawings

[0011] The invention will now be described with reference to the drawings in which:

[0012] Figure 1, is a prior art diagram of directory files within a IC smart card as well as a prior art method of encoding data within the smart card; and

[0013] Figure 2, illustrates an improved method of encoding data within a smart card resulting in a single continuous block of free memory.

Detailed Description of the Invention

[0014] In prior art Figure 1, a PKCS15 compatible integrated circuit smart card (SC) is shown. The smart card comprises of a Master Directory File (MF) 10, with a pointer to an Object Directory File (ODF) 11. The ODF 11 contains pointers to data directory files stored on the smart card, namely a Private Key Directory File (PrKDF) 12, Public Key Directory File (PuKDF) 13, Secret Key Directory Files (SKDF) 14, Certificate Directory File (CDF) 15, and Data Object Directory File (DODF) 16. Memory provided to each of these directory files is non-volatile, byte addressable, having a start address 22 and an end address 24.

[0015] Memory for each data directory file 17 comprises of a pointer memory space 18 and a pointer data memory space 20. The pointer memory space 18 is reserved for encoding pointer addresses, and the pointer data memory space 20 reserved for encoding pointer data. Information space within memory in proximity of the smart card start address 22 is reserved for pointers 18. Information space within the memory in proximity

of a pointer data start address 23 is reserved for pointer data 20. The pointer data 20 entries are referenced by a corresponding pointer 18, so for instance the pointer stored in pointer memory space "A" references a memory address corresponding to pointer data "A" stored in the pointer data memory. In this manner any references made to the pointer data "A" are indexed within the pointer memory by pointer "A".

[0016] Pointers are sequentially encoded into the pointer memory space 18 starting in proximity of the memory start address 22, and pointer data is encoded starting at the pointer data start address 23. As information is encoded into the smart card directory file memory 17 the amount of unused memory decreases, resulting in two separate blocks of memory having no data contained therein. A first block of memory 19 having no data contained therein is located between the last pointer entry, entry "C", written into the pointer memory space 18 and the pointer data start address 23. A second block of memory 21 having no data contained therein is located between the last pointer data entry "C", written into the pointer data memory space 20, and the memory end address 24. The result is two unused memory blocks having no data contained therein. Of course, it is understood that the unused memory is being reserved for future uses when more data is stored.

[0017] A location of the pointer data start address 23 is a design issue for each of the data directory files 17 in order to make the best use of the memory available to the directory file. During encoding of information within the directory file 17 the prior art encoding method limits the encoding to a fixed number of pointers or to a fixed amount of pointer data, which must be predetermined before encoding, due to the placement of the pointer data start address prior to encoding. In either case this results in two non-sequential blocks of memory within the directory file 17 having no data encoded therein.

[0018] In designing the memory storage map for a smart card compliant with PKCS15, there are competing concerns. These are illustrated by way of example. When storing private keys in a data area, space allocated to pointers is based on a theoretical maximum number of keys. When all keys are identical in configuration, a simple mathematical calculation allows for a determination of the pointer data start address 23.

Here, the memory available divided by the memory required for each key object plus the memory required for each pointer determines the maximum number of keys that can be stored. This also determines the location of the pointer data start address 23 which is at least the maximum number of pointers multiplied by the amount of space each pointer requires from the beginning of the data directory file. Unfortunately, when more than one key size is supported, the calculation is not so simple. Either the maximum number of keys is supported - memory usage is optimized only when all keys are of a same smallest key size – or fewer keys are supported such that there is a potential that pointer space will run out before data object space. Of course, in the first instance, storing of larger keys results in unused pointer locations.

[0019] Problematically, because the PKCS15 standard is already widely implemented and accepted, it is difficult at present to redesign the data storage within a smart card. Improving a flawed standard requires acceptance by the standards body and its participants. This is an onerous task that sometimes requires years to achieve. An alternative to such an approach is to design a backwards-compatible improved storage system. Unfortunately, such an approach severely restricts available design options and is often futile.

[0020] A backwards-compatible improved storage system must be such that reading thereof functions on any PKCS15 compliant smart card reading device. Writing of the smart cards need not follow the PKCS15 standard exactly, as long as the data is retrievable by PKCS15 systems.

[0021] One such improvement over the prior art method of encoding data within the smart card is shown in Figure 2. In Figure 2, a single directory file (DF) 25 is shown. The memory provided to the directory file has a start address 28 and an end address 29. Within each data directory file there is a pointer memory space 31 and an object data memory space 33. The pointer memory space 31 is reserved for storing pointer addresses, and the object data memory space 33 reserved for storing pointer data. Space within memory in proximity of the start address 28 is reserved for storing pointers 31. Space within the memory in proximity of the end address 29 is reserved for pointer data